Docket No. 50277-2416

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Confirmation No. 8375
Zhen Liu, et al.	Group Art Unit No.: 2166
Serial No.: 10/810,152	Examiner: Navneet K. Ahluwalia
Filed: March 26, 2004	

For: TECHNIQUES FOR MANAGING XML DATA ASSOCIATED WITH MULTIPLE EXECUTION UNITS

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF ON APPEAL

Sir:

Further to the Notice of Appeal filed August 7, 2009, and in reply to the Examiner's Answer mailed on January 5, 2010, the Appellants hereby submit their Reply Brief on appeal pursuant to 37 C.F.R. § 41.41.

The shortened statutory period for the Reply Brief runs until March 5, 2010. The Appellants respectfully request reconsideration of the application in light of the following remarks. Each of the new arguments advanced in the Examiner's Answer is addressed below.

FIRST ARGUMENT

For the first time in the application history, the Office cites column 2 (ll. 16-28), column 6 (ll. 61-67), column 37 (ll. 31-40), column 36 (ll. 25-35) and column 35 (ll. 42-48) of *Fernandez* as allegedly disclosing "detecting that a portion of a query execution plan… will cause a first producer execution unit, that will perform said portion, to generate **XML data for**

use by a second consumer execution unit in performing another portion of said query execution plan," recited in Claim 1. (Examiner's Answer: page 14 (ll. 5-17))

However, these portions of *Fernandez* differ in many fundamental ways from the limitation against which the portions are cited.

In column 2 (ll. 16-28), *Fernandez* describes an XML view that is used to generate an XML output document when execution of all SQL queries of the execution plan is completed. However, the XML view is not passed from one execution unit to another execution unit during performance of the execution plan, as claimed.

In column 6 (ll. 61-67), *Fernandez* describes a query composer that composes an RXL query from a user XML query and a virtual view of the database. However, the composer does not execute any portion of a query execution plan, and thus neither the XML query nor the virtual view are the XML data that is passed **from one execution unit to another execution unit** during performance of the query execution plan, as claimed.

In column 37 (ll. 31-40), *Fernandez* describes that a query composer uses a virtual view and an RXL query to generate subtrees for the RXL query, and that a translator generates SQL queries for the subtrees. However, neither the query composer nor the translator executes any portion of a query execution plan. Further, none of the subtrees, the RXL query and the SQL queries are the XML data that is passed **from one execution unit to another execution unit** during performance of the query execution plan, as claimed.

In column 36 (ll. 25-35), *Fernandez* describes that an RXL query can be represented as a view tree, in which each node corresponds to an element in one of the construct clauses in the RXL query and is annotated by a non-recursive datalog query that computes all instances of that node in the output XML document. The mapping captures relationships between deeply nested variables in the RXL query and the variables in the SQL queries. However, the relationships are used after all SQL queries of the RXL query finish generating and providing their results. The relationships are not "**for use by a second consumer execution unit in performing another portion of said query execution plan**," as recited in Claim 1.

In column 35 (ll. 42-48), *Fernandez* describes a mapping between deeply nested variables in an RXL query and variables in SQL queries. *Fernandez* states that one or more SQL queries are computed to extract and group the data for the XML view, and that XML tags, representing the mapping between the nested variables in the user RXL query and the variables in the SQL queries are used to generate an output XML document. However, neither the XML view nor the XML tags are "**for use by a second consumer execution unit in performing another portion of said query execution plan**," as recited in Claim 1.

Therefore, none of the excerpts that the Office cites for the first time demonstrates the limitation for which the excerpts are used, namely that "a **first producer execution unit, that will perform said portion, ... to generate XML data for use by a second consumer execution unit in performing another portion of said query execution plan**," as recited in Claim 1.

SECOND ARGUMENT

For the first time in the application history, the Office cites column 37 (ll. 31-40) and column 38 (ll. 13-23) as allegedly disclosing **two execution units** executing portions of a query execution plan, where the second unit uses the XML data generated by the first unit as recited in Claim 1. (Examiner's Answer: page 15 (ll. 6-9))

These portions of *Fernandez* provide examples of generating query execution plans by splitting a view tree of an RXL query into a collection of subtrees and associating SQL queries with the subtrees. Each plan may include between 1 and 10 tuple streams to produce XML elements. Each SQL query is submitted to an RDBMS, which returns data that is merged to produce an output XML document. However, nowhere does *Fernandez* state that *Fernandez* has a **first execution unit involved in performing one portion of a query and that the first execution unit feeds the obtained results to a second execution unit involved in performing another portion of the same query**, as recited in Claim 1.

THIRD ARGUMENT

For the first time in the application history, the Office alleges that, in column 35 (ll. 42-48), *Fernandez* discloses that *Fernandez'* query **execution plan returns XML data**, as recited in Claim 1, not relational data tuples. (Examiner's Answer: page 15, ll. 13-15)

In column 35 (ll. 42-48) *Fernandez* describes that to evaluate an RXL query, one or more SQL queries are computed by an RDBMS to extract and group data for the XML view and then XML tags are added to the grouped data. The XML tags are added by an XML generator (also referred to as an XML integrator), which generates an output XML document. The XML tags capture the nested relations between the variables in the RXL query so that the nested relationships can be created (restored) in the output XML document. However, the tags are generated by a query composer, not by the RDBMS. *Fernandez'* RDBMS **generates data tuples** for the SQL queries, **not XML data for use by another execution unit in executing another portion of the same SQL query**, as claimed.

FOURTH ARGUMENT

For the first time in the application history, the Office alleges that in column 37 (ll. 31-40) and in column 38 (ll. 13-23), *Fernandez* describes how a query execution plan is split so that the result of one is being used by the next execution unit to continue the process "**to get the result**." (Examiner's Answer: page 16, ll. 3-5)

In the cited excerpts, *Fernandez* describes that from a user XML query, a query composer generates an RXL query and generates a mapping of the nested variables in the user XML query. From the RXL query, *Fernandez'* translator generates an execution plan and SQL queries. The execution plan and the SQL queries are sent to an RDBMS, which executes the execution plan (and the SQL queries). The output from the RDBMS, i.e. data tuples for the SQL queries, is sent to an XML generator. The XML generator merges the data tuples according to the mapping of the nested variables and generates an output XML document. Thus, in *Fernandez,* only RDBMS executes SQL queries of the execution plan. None of other modules in *Fernandez*, i.e., the query

composer, the translator or the XML generator executes a portion of the execution plan for the same query, as claimed. In *Fernandez,* each of the modules "gets the result;" however, none of the results of one module is used "by a **second consumer execution unit in performing another portion of said query execution plan**," as recited in Claim 1.

FIFTH ARGUMENT

On page 16 (ll. 10-17), the Examiner states "in response to Appellant's arguments Examiner submits that *Fernandez* teaches the annotating said information with an annotation that causes XML data generated by said execution unit Examiner would like to acknowledge the examples and explanation given as to regards with what the invention is in the examples and explanation given as to regards with what the invention is in the response."

The applicants are unsure how to reply to the above statement because it is unclear what the statement means. Further explanation of the statement is required because without any explanation, the statement seems to be irrelevant and out of place.

SIXTH ARGUMENT

For the first time in the application history, the Office alleges that in column 37 (ll. 31-40) and column 38 (ll. 13-23), *Fernandez* describes "wherein the step of generating information includes generating information that, prior to **annotating** said information, would cause said first execution unit to generate said XML data in a first form that cannot be used by said second execution unit," recited in Claim 2.

In column 37 (ll. 31-40), *Fernandez* describes generating a view tree of an RXL query, subtrees from the RXL query and SQL queries for the subtrees. In column 38 (ll. 13-23) *Fernandez* describes that each SQL query is submitted to an RDBMS, which returns data that is merged to produce an output XML document. However, nowhere does *Fernandez* state that *Fernandez* has one execution unit, which, prior to annotating said information, would cause said first execution unit to generate XML data in a first form that cannot be used by a second

execution unit, as recited in Claim 2.

The Office seems to treat *Fernandez'* query composer, query translator and XML document generator as execution units recited in Claim 2. This is incorrect because none of *Fernandez'* query composer, query translator and XML document generator executes portions of a query execution plan, as recited in Claim 2. Due to the fact that Claim 2 depends from Claim 1, Claim 2 recites definitions of execution units included in Claim1, which recites that the first producer execution unit performs one portion of a query execution plan and generates XML data for use by a second consumer execution unit that performs another portion of the same query execution plan. In *Fernandez* the RDBMS executes the execution plan of SQL queries that are submitted to it. To the extent that those execution plans are executed by multiple execution units, they would have to be multiple execution units **within** the RDBMS. However, *Fernandez* does not disclose any detail about how those SQL statements are executed within the RDBMS, and *Fernandez'* RDBMS itself does not meet the requirements for the execution units set forth in Claim 1. Further, none of *Fernandez'* query composer, query translator and XML document generator meets the requirements for the execution units set forth in Claim 1.

Therefore, none of the excerpts that the Office cites demonstrates the limitations expressly set forth in the claims.

CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejections of Claims 1-50 lack the requisite factual and legal bases. Appellants respectfully request that the Honorable Board reverse the rejections of Claims 1-50.

For the reasons indicated above, all pending claims present subject matter that is patentable over the references of record, and are in condition for allowance. Therefore, Applicants respectfully request reversal of the final rejections expressed in the Office Action and reiterated in the Examiner's Answer.

Throughout the pendency of this application the Commissioner is hereby authorized to charge any applicable fee, including extension of time fees, and to credit any overpayment to our Deposit Account No. 50-1302.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP

Dated: February 25, 2010

/MalgorzataAKulczycka#50496/
Malgorzata A. Kulczycka
Reg. No. 50,496

2055 Gateway Place, Suite 550
San Jose, California 95110-1089
Telephone: (408) 414-1228
Facsimile: (408) 414-1076